САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**КАФЕДРА «КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И СИСТЕМ»**

**Арзуманян Наринэ Карапетовна**

**Выпускная квалификационная работа бакалавра**

# Анализ и разработка рекомендательной системы

Направление 010400

Прикладная математика и информатика

Научный руководитель,
доктор физ.-мат. наук,
профессор
Веремей Е. И.

Санкт-Петербург

2017

## Содержание

# Введение

В настоящее время нас окружает огромное количество товаров и услуг, а также большие объемы информации, на обработку которой уходит много времени. Такое разнообразие объектов имеет свои плюсы и минусы: с одной стороны, каждый человек может выбрать наиболее подходящий ему объект; с другой стороны, выбрать становится сложнее. Какие же методы используют люди, чтобы все-таки выбрать новый объект среди такого множества?

Например, можно спросить совета у другого человека: друга, специалиста в этой области, консультанта, продавца. Но такой метод нельзя назвать быстрым и точным, ведь рекомендация основана на личном мнении другого человека. Если не задействовать других людей, как же облегчить себе задачу и прийти к нужному решению?

Можно добавить ограничения на поиск, т.е. указать конкретные характеристики, которыми должен обладать объект, тем самым сузив количество рассматриваемых объектов. Такой процесс, безусловно, сужает поиск, но число объектов все равно может остаться очень большим. Задавать «более грубые» значения характеристик (например, узкий диапазон стоимости объекта, точный год выпуска и др.) или определять больший перечень фиксированных свойств (например, размеры, материал и др.), тем самым максимально сужая поиск, не всегда является возможным. Во-первых, при поиске нового объекта человек сам не всегда четко понимает, что же он ищет; во-вторых, сужение поиска в принципе не всегда является нужным.

Так появляется необходимость в построении системы, которая будет выделять объекты не только по фиксированным свойствам, но и с использованием дополнительных данных. Поэтому создаются так называемые рекомендательные системы, которые используют различные принципы построения рекомендаций в зависимости от поставленной задачи.

Основу рекомендательных систем составляют программы, которые формируют рекомендации для пользователей исходя из определенной информации о возможной системе их предпочтений, а также исходя из информации об объектах. Такие системы используются, например, в процессе выбора нового фильма, музыки или книги. Рекомендательные системы полезны не только их непосредственным пользователям, но и владельцам объектов, среди которых пользователь производит выбор. Действительно, рекомендательные системы широко используются, например, в интернет-магазинах, где точность прогноза особенно важна. Чем точнее сформирована рекомендация, тем довольнее останется клиент, что может увеличить прибыль компании. Также рекомендательные системы помогают увеличить объем продаж: предлагают сопутствующие товары или товары из другой области, но которые могут заинтересовать данного пользователя исходя из его предпочтений.

Многие известные по всему миру сайты уже используют рекомендательные системы: Ozon, eBay, Amazon, Кинопоиск, IMDb, Pandora и др., но ни одна из систем не может гарантировать 100% точность сформированной рекомендации. Методы построения прогноза нуждаются в усовершенствовании, а для этого целесообразно провести сравнительный анализ различных типов существующих рекомендательных систем.

Таким образом, разработка рекомендательных систем является актуальной задачей в настоящее время, в связи с чем и была выбрана темой для данной работы. Анализ различных типов рекомендательных систем и методов построения рекомендаций также является важным предметом для исследования, так как выбор метода в конкретной задаче позволяет увеличить точность рекомендаций.

Данная работа связана с вопросами формирования рекомендательных систем для выбора фильмов. В работе рассматривается группа пользователей и набор фильмов, а также оценки фильмов этими пользователями. Учитывая

оценки, которые конкретный пользователь поставил части фильмов, прогнозируется его оценка фильмам, которые он ранее не оценивал. Оценка формируется с использованием двух разных подходов: методом k-ближайших соседей с различными мерами сходства пользователей, а также методом приближения матрицы оценок с использованием сингулярного разложения матрицы с различным рангом приближенной матрицы.

# Постановка задачи

Пусть имеется $n$ пользователей и $m$ фильмов. Каждый пользователь ставит оценки произвольному числу фильмов, таким образом формируется матрица оценок $R$ размерности $n \times m$. Элемент $r_{i,j}$ этой матрицы соответствует оценке, данной $i$-м пользователем $j$-ому фильму. Все оценки являются целыми числами, принадлежащими отрезку [1; 5]. Далее матрица $R$ дозаполняется нулями, если $i$-й пользователь не поставил оценку $j$-ому фильму.

В каждой строке матрицы $R$ нулевые элементы характеризуют фильмы, которым данный пользователь не поставил оценку. Одной из центральных задач рекомендательной системы является поиск возможных (прогнозных) оценок для исходно нулевых элементов, ведь чтобы порекомендовать пользователю некоторый фильм, необходимо убедиться, что ему этот фильм понравится, т.е. он сам поставил бы высокую оценку этому фильму. Таким образом, в данной работе ставится задача поиска прогнозных оценок изначально пропущенных (изначально нулевых) оценок фильмам различными методами, а также задача сравнения точности таких прогнозов, их полноты и времени построения рекомендаций.

Пусть для некоторого пользователя рекомендательной системой была получена оценка $\widehat{r_{i,j}}$, в то время как реальная оценка существовала и была равна $r_{i,j}$. Пусть было получено $num\_it$ оценок и все соответствующие реальные оценки существовали. Ошибка прогноза будет считаться как нормированная средняя абсолютная погрешность:

$$err = \sum_{j=1}^{num\_it} \frac{|\widehat{r_{i,j}} - r_{i,j}|}{5 * num\_it} \cdot 100\%.$$

Точность прогноза (*precision*) считается в процентах с использованием вышеописанной ошибки: $prc = 100\% - err$.

Полнота прогноза (*recall*) представляет собой соотношение количества оценок, которые удалось построить при использовании определенного метода, к общему числу прогнозируемых оценок.

Так как в задаче рекомендации фильмов точность очень важна, введем дополнительное условие на значение ошибки. Пусть в среднем абсолютная разность прогнозной и реальной оценок равна единице, тогда значение ошибки будет 20%. Добьемся того, чтобы алгоритм чаще определял точное значение, чем ошибался, т.е. в среднем ошибался меньше, чем на одну единицу, тогда ошибка прогноза должна быть меньше 20%.

Математическая постановка задачи: $\min\limits_{err<20\%}(\alpha * \mathrm{err} + \beta * \mathrm{t})$,

где $err -$ вышеописанная ошибка, $t -$ время построения прогноза, $\alpha, \beta -$ коэффициенты, которые определяются исходя из условий конкретной задачи и с учетом нормировки.

# Обзор литературы

В настоящее время рекомендательные системы представляют собой широко обсуждаемую и изучаемую тему. Каждый подход, взятый за основу в конкретной рекомендательной системе, нуждается в доработке для достижения максимально хорошего результата. При написании данной работы использовались научная и учебно-методическая литература по вопросам коллективного разума, построения рекомендательных систем, а также по алгебраическим методам.

Теоретические основы построения рекомендаций и существующие подходы к построению прогнозных оценок рассматриваются в основном из книги Тоби Сегарана «Программируем коллективный разум» [1], книги «Recommender Systems Handbook» нескольких авторов [2] и статьи «Анализ алгоритмов обучения коллаборативных рекомендательных систем» [3]. В этих источниках подробно описано, что такое рекомендательные системы, зачем они нужны и какие они бывают. После изучения основных методов построения рекомендательной системы, было выбрано два подхода, которые применяются к построению системы, рекомендующей выбор фильмов.

Первый рассматриваемый подход – метод k-ближайших соседей, описанный в лекциях по метрическим алгоритмам классификации для произвольных объектов [4]. После его адаптации к задаче рекомендации фильмов поиск наиболее похожих пользователей в этой выпускной квалифицированной работе стал производиться только среди пользователей, оценивших фильм, для которого строится оценка.

Второй подход – метод сингулярного разложения матрицы без использования базовых предикторов, описанный в статье «Анализ алгоритмов обучения коллаборативных рекомендательных систем» [3]. В поставленной задаче оценки, полученные путем приближения матрицы R, обязательно должны принадлежать отрезку [1; 5], ведь пользователи могли

ставить оценки только из этого диапазона. Таким образом, адаптированный к этой задаче алгоритм SVD должен изменять вышедшие за границы нужного отрезка оценки до попадания в него.

Используемые определения из алгебры, в частности, определение сингулярного разложения, а также важная в рассматриваемой задаче теорема Эккарта-Янга, взяты из [5], [6] и [7].

# Глава 1. Особенности известных рекомендательных систем

Рекомендательные системы – программные средства, которые предсказывают, какие объекты будут интересны пользователю с учетом информации о нем и объектах [3]. Такие данные могут быть получены как явными, так и неявными способами. К явным относят следующие методы: пользователь ставит оценку объектам из заданного диапазона или ранжирует группу объектов, выбирает наиболее привлекательный из двух (небольшой группы) объектов или указывает топ-лучших/худших объектов и др. К неявным методам относятся: число просмотров объекта (например, сколько раз была прослушана аудиозапись), категории просматриваемых объектов (например, в интернет-магазинах, где есть точное разделение на категории) и др.

В рекомендательных системах можно выделить 4 основных типа [2], [8], описанных далее.

## 1. Системы, основанные на контенте

В такого рода рекомендательных системах используются так называемые профили пользователей и характеристики объектов. Каждому объекту ставятся в соответствие некоторые значения параметров-характеристик, на основе которых производится прогноз оценки объекта с учетом интересов пользователя исходя из его профиля. При использовании такого подхода рекомендуются объекты, которые похожи по характеристикам с объектами, понравившимися пользователю.

Когда новый пользователь, у которого пустой профиль, начинает использовать рекомендательную систему, основанную на контенте, ему необходимо дать некоторую информацию о себе. Это можно сделать разными способами: путем задания важных (предпочтительных)

значений характеристик, указанием интересных ему объектов (при сравнении которых можно будет выделить ключевые характеристики для этого пользователя), а также добавлением личной информации (возраст, пол, место проживания и т.д.).

При использовании рекомендательных систем такого типа основной проблемой, с которой сталкиваются разработчики, является проблема нахождения значений параметров-характеристик объекта, так как количество таких характеристик может быть очень большим (например, в подходе Music Genom Project используется 400 характеристик, которые описывают музыкальные треки [9]). Задавать такое количество характеристик каждому объекту вручную очень сложно и долго, также такие значения характеристик оказываются субъективными, ведь разные эксперты могут поставить разное значение некоторым характеристикам.

Преимуществом основанных на контенте рекомендательных систем является отсутствие проблемы «холодного» старта (ситуация, когда объекту еще никто не поставил никакую оценку или таких оценок слишком мало). При использовании этого метода пользователям достаточно заполнить хотя бы частично свои профили, после чего им можно рекомендовать как новые, так и давно имеющиеся в базе и уже рекомендованные другим пользователям объекты.

## 2. Системы с коллаборативной фильтрацией

Методы коллаборативной фильтрации в свою очередь можно разделить на 3 типа: основанные на соседстве, основанные на модели и гибридные.

Методы коллаборативной фильтрации, основанные на соседстве, рекомендуют пользователю объекты на основании оценок похожих пользователей. Меру сходства пользователей можно определять разными способами, часть из которых будет рассмотрена далее. Метод

*k*-ближайших соседей может основываться как на сходстве пользователей (*user-based*), так и на сходстве объектов (*item-based*).

Методы коллаборативной фильтрации, основанные на модели, находят некоторые закономерности в оценках пользователей. К такому типу рекомендательных систем относят метод сингулярного разложения матриц, метод байесовских сетей, кластеризации, латентной семантической модели и др. [10].

Главным недостатком методов коллаборативной фильтрации является наличие проблемы «холодого» старта. При использовании этого подхода новые объекты не будут рекомендоваться пользователям в силу отсутствия оценок этого объекта.

Тем не менее, использование предыдущих оценок активного пользователя (для которого строится рекомендация) и построение оценок с учетом оценок группы наиболее похожих на активного пользователя людей является одним из самых популярных способов построения рекомендаций в силу своей сравнительной простоты и хороших результатов.

### 3. Системы, основанные на знаниях

Такого рода методы рекомендации используют конкретные предметные знания об объектах и том, как эти объекты отвечают предпочтениям и нуждам пользователей. Основанные на знаниях системы используют дополнительную информацию о группе объектов в целом. Например, если человек в интернет-магазине покупает фотоаппарат, то к нему можно сразу предложить купить чехол. Такой подход дает хорошие результаты в задачах, когда рекомендовать похожие товары не целесообразно (например, при покупке квартиры вряд ли человек будет сразу же выбирать еще одну, а вот мебель, техника и т.д. ему может пригодиться).

В основанных на знаниях рекомендательных системах можно также выделить на 2 типа: ориентированные на ситуацию и ориентированные

на ограничения. Ориентированные на ситуацию рекомендательные системы строят рекомендации, опираясь на меру сходства нужд пользователя (описания проблемы) и возможной рекомендации (решения проблемы). Ориентированные на ограничения рекомендательные системы преимущественно используют предопределенные базы знания о том, как соотнести потребности пользователя с характеристиками объектов.

Построение системы знаний о группе объектов является сложной задачей, решаемой для каждой конкретной группы объектов отдельно, что является как недостатком метода (трудоемкий), так и преимуществом (позволяет построить максимально подходящую под конкретную задачу рекомендацию).

4. Гибридные системы

Гибридные рекомендательные системы сочетают в себе несколько различных подходов рекомендации. При комбинации двух методов слабые стороны одного метода исправляют сильными сторонами второго метода, чтобы методы дополняли друг друга и решали определенные задачи. Например, можно использовать гибридную рекомендательную систему, сочетающую методы, основанные на контенте (для решения проблемы холодного старта), и коллаборативую фильтрацию (для выявления скрытых предпочтений пользователей).

Такое разнообразие рекомендательных систем позволяет в каждой конкретной ситуации выбрать наиболее подходящий метод с учетом особенностей задачи. Действительно, в некоторых задачах пользователю необходимо рекомендовать объекты, обладающие некоторыми характеристиками, которые важны и интересны пользователю (тогда целесообразно использовать методы, основанные на контенте); в других ситуациях похожие друг на друга объекты пользователю точно не будут интересны и лучше советовать ему сопутствующие объекты (тогда

используются методы, построенные на знаниях); в ситуациях, когда известно достаточно большое количество оценок системы пользователей, а также если активный пользователь (тот, для которого строится рекомендация) указал достаточное число оценок различным объектам, пользователю могут быть интересны объекты, которые также интересны группе пользователей, имеющих схожие оценки других объектов (тогда целесообразно применять методы коллаборативной фильтрации и при выполнении условия достаточности оценок, результат будет довольно хорошим). Достаточное число оценок определяется в каждом конкретном случае индивидуально.

Безусловно, при правильной комбинации различных методов гибридные системы очень часто дают еще более точные прогнозы, чем составляющие их методы, но при этом увеличивается время построения рекомендации, что не всегда целесообразно.

# Глава 2. Методы построения рекомендаций

## 2.1. Метод *k*-ближайших соседей

Метод *k*-ближайших соседей относится к метрическим алгоритмам классификации, основанным на оценивании сходства объектов [4]. Активному пользователю (для которого строится рекомендация) прогнозируется оценка для некоторого объекта как взвешенная сумма оценок *k* наиболее похожих на него пользователей.

Рассмотрим двух пользователей $(p_1,\ p_2)$ из множества всех пользователей $P$ с соответствующими оценками $(r_1,\ r_2)$ и введем функцию $\mu$ меры сходства этих пользователей:

$$\mu: P \times P \ \rightarrow \mathbb{R},$$

где каждый пользователь $p \in P$ характеризуется вектором своих оценок

$r \in R: r = (r_1, \dots, r_m)$.

Для удобства зададим дополнительное условие для функции меры: значение меры должно принадлежать отрезку [0; 1], причем чем больше значение меры, тем больше пользователи похожи, т.е. значение 1 соответствует полному совпадению интересов пользователей. Мера сходства пользователей может определяться разными способами, в работе используются три подхода [1], [11], [12]:

1) Мера Эвклида

Каждый пользователь характеризуется вектором своих оценок и представляет собой точку на $m$-мерной плоскости ($m$ – количество используемых фильмов). Тогда получается, что чем ближе находятся пользователи, тем больше у них похожих оценок и тем больше совпадают их интересы. Найдем эвклидово расстояние между двумя пользователями:

$$\rho(p_1, p_2) = \sqrt{\sum_{i=1}^{m} (r_{1,i} - r_{2,i})^2}.$$

Значение эвклидова расстояния принадлежит отрезку $[0, +\infty)$, поэтому в мере Эвклида расстояние используется следующим образом:

$$\mu(p_1, p_2) = \frac{1}{1+\rho(p_1,p_2)}.$$

Если пользователи похожи (поставили одинаковые оценки), то эвклидово расстояние между пользователями будет равно нулю, а мера Эвклида – единице. Если интересы пользователя очень разные и расстояние между пользователями большое, то значение меры Эвклида будет стремиться к нулю, что соответствует установленному дополнительному условию для функции меры.

2) Косинусная мера

Еще одним методом определения сходства пользователей является косинус угла между радиус-векторами соответствующих этим пользователям точек на $m$-мерной плоскости. Стоит заметить, что в рассматриваемой задаче все точки, соответствующие пользователям, лежат в первой четверти, так как все оценки фильмов положительные по условию. Значение косинуса уже принадлежит отрезку [0; 1], причем если сравнить пользователя с самим собой (угол равен нулю), то косинус будет равен 1, что соответствует полному совпадению интересов. Чем больше угол между двумя векторами (при этом угол меньше либо равен 90º), тем меньше значение косинуса и тем меньше пользователи похожи друг на друга, т.е. дополнительное условие для функции меры так же выполнено. Таким образом, косинусная мера находится следующим образом:

$$\mu(p_1, p_2) = \cos(p_1, p_2) = \frac{r_1 \cdot r_2}{\|r_1\| \cdot \|r_2\|}.$$

3) Мера Пирсона

Следующий рассматриваемый в этой работе метод определения степени сходства пользователей основан на коэффициенте корреляции

Пирсона, который показывает, существует ли линейная связь между двумя показателями. Действительно, есть пользователи, которые «занижают» оценки всем фильмам, и пользователи, которые ставят «завышенные» оценки. Если сравнивать непосредственно сами значения оценок, то эти два пользователя не будут похожи друг на друга, но на самом деле, это не так. Если разница между оценками двух пользователей постоянна, то это как раз говорит о том, что пользователи имеют схожие вкусы, что и покажет коэффициент корреляции Пирсона. Формула для нахождения коэффициента корреляции Пирсона:

$$r(p_1, p_2) = \frac{cov(r_1, r_2)}{\sqrt{s_{r_1}^2 \cdot s_{r_2}^2}} = \frac{\sum_{i=1}^{m}(r_{1,i} - \overline{r_1})(r_{2,i} - \overline{r_2})}{\sqrt{\sum_{i=1}^{m}(r_{1,i} - \overline{r_1})^2 \sum_{i=1}^{m}(r_{2,i} - \overline{r_2})^2}},$$

где $\overline{r_1}$, $\overline{r_2}$ – средние арифметические оценок пользователей $p_1, p_2$ соответственно.

Значение коэффициента корреляции Пирсона принадлежит отрезку [-1; 1], абсолютное значение, равное 1, соответствует наличию линейной связи между пользователями и схожести их предпочтений, а значение 0 – отсутствию связи, что характерно для непохожих вкусов пользователей. Чтобы мера Пирсона соответствовала дополнительному условию (значение меры принадлежало отрезку [0; 1]), необходимо в качестве меры принять абсолютное значение коэффициента корреляции Пирсона:

$$\mu(p_1, p_2) = |r(p_1, p_2)|.$$

В соответствии с описанным методом *k*-ближайших соседей сформируем вычислительный алгоритм для определения новой оценки фильма с идентификатором *id_f* для активного пользователя с идентификатором *id_p*:

1) Задать параметр *k*.

2) Из пользователей, оценивших фильм с идентификатором *id_f*, найти *k* наиболее похожих пользователей для активного пользователя с идентификатором *id_p*.

3) Определить оценку пользователя *id_p* фильму *id_f* как округленную взвешенную сумму наиболее похожих на него пользователей:

$$\hat{r}_{\text{id\_p,id\_f}} = round\left(\frac{\sum_{i=1}^{k} r_{i,id\_f} \cdot \mu(id\_p,i)}{\sum_{i=1}^{k} \mu(id\_p,i)}\right).$$

Используется округленная взвешенная сумма, так как все оценки являются целыми числами от 1 до 5.

## 2.2. Метод SVD

Прежде чем приступить к описанию метода, приведем основные теоретические сведения по рассматриваемой теме.

*Определение.* Невырожденная квадратная матрица $A$ называется ортогональной, если для нее верно соотношение [13]:

$$A^{-1} = A^{T}.$$

*Определение.* Сингулярным разложением матрицы $A \in \mathbb{R}^{m \times n}$ называется ее представление в виде произведения трех матриц:

$$A = Q\Sigma P^{*},$$

где $P$ – ортогональная матрица порядка $n$, $Q$ - ортогональная матрица порядка $m$, $\Sigma$ - диагональная матрица, на диагонали которой стоят сингулярные числа матрицы $A$ в порядке убывания [5], [6], [7].

*Определение.* Нормой матрицы с действительными или комплексными элементами является действительное число $\|A\|$, т.е. отображение $\|A\|: \mathbb{R}^{n\times m}(\mathbb{C}^{n\times m}) \to \mathbb{R}$, удовлетворяющее аксиомам [5]:

1. $\|A\| \geq 0$ (неотрицательность),
2. $\|\alpha A\| = |\alpha| \cdot \|A\|$ (однородность),
3. $\|A + B\| \leq \|A\| + \|B\|$ (неравенство треугольника),
4. $\|AB\| \leq \|A\| \cdot \|B\|$ (мультипликативность)

*Определение.* Фробениусовой нормой матрицы является [5]:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}\left|a_{i,j}\right|^2}.$$

*Теорема (Эккарт-Янг).* Пусть дана матрица $A$ размерности $n\times m$, для которой известно сингулярное разложение $A = Q\Sigma P^*$ и которую требуется аппроксимировать матрицей $A_k$ с заданным рангом $k < r = rank(A)$. Если в матрице $\Sigma$ оставить $k$ наибольших значений сингулярных чисел, а остальные заменить нулями, то разложение $A = Q\Sigma_k P^*$ даст наилучшее приближение исходной матрицы в том смысле, что фробениусова норма разности матриц будет минимальна [6].

Приближение матрицей меньшего ранга позволяет снизить шум (информация, которая не несет в себе полезных данных и затрудняет прогнозирование оценок) и разреженность (количество нулевых элементов) исходной матрицы [6], что будет полезно в задаче рекомендации фильмов.

Возвращаясь к поставленной задаче, представим сингулярное разложение матрицы оценок $R$ в виде произведения трех матриц:

$$R = UDV,$$

где размерность матрицы $R$ - $n\times m$, $U$ - $n\times n$, $D$ - $n\times m$, $V$ – $m\times m$, а диагональные элементы матрицы $D$ упорядочены по убыванию.

Размерность матрицы $R$ является достаточно большой в рассматриваемой задаче ($963\times1682$), но при этом матрица является сильно разреженной, так как пользователи ставили оценки далеко не всем фильмам и в матрице $R$ всего 100000 оценок (ненулевых элементов). Пусть $r$ – ранг

матрицы $R$, аппроксимируем матрицу $R$ матрицей $\tilde{R}$ ранга $k < r$. С учетом теоремы Эккорта-Янга получим следующее приближение:

$$\tilde{R} = UD_kV,$$

где в матрице $D_k$ сохранены первые $k$ сингулярных чисел матрицы $D$, а остальные заменены нулями. Так как нулевые строки матрицы $D_k$ не несут в себе никакой информации (последние $n$-$k$ строк являются нулевыми), а при умножении матрицы $U$ на $D_k$ последние $n$-$k$ столбцов не будут влиять на результат умножения (так как эти элементы будут умножаться на нули), то размерность матриц $U, D_k$ и, по аналогии, матрицы $V$ можно уменьшить, не влияя на результирующую матрицу $\tilde{R}$. Таким образом, матрицу $R$ можно приблизить матрицей $\tilde{R}$, представимой в виде произведения трех матриц:

$$\tilde{R} = \tilde{U}\tilde{D}\tilde{V},$$

где размерность матрицы $\tilde{R}$ - $n \times m$, $\tilde{U}$ - $n \times k$, $\tilde{D}$ - $k \times k$, $\tilde{V}$ – $k \times m$ (рис.1).

Преобразовав эту аппроксимацию и представив ее в виде произведения двух матриц, получим:

$$\tilde{R} = \hat{U}\tilde{V},$$

где $\hat{U} = \tilde{U}\tilde{D}$, а размерность $\hat{U}$ - $n \times k$.

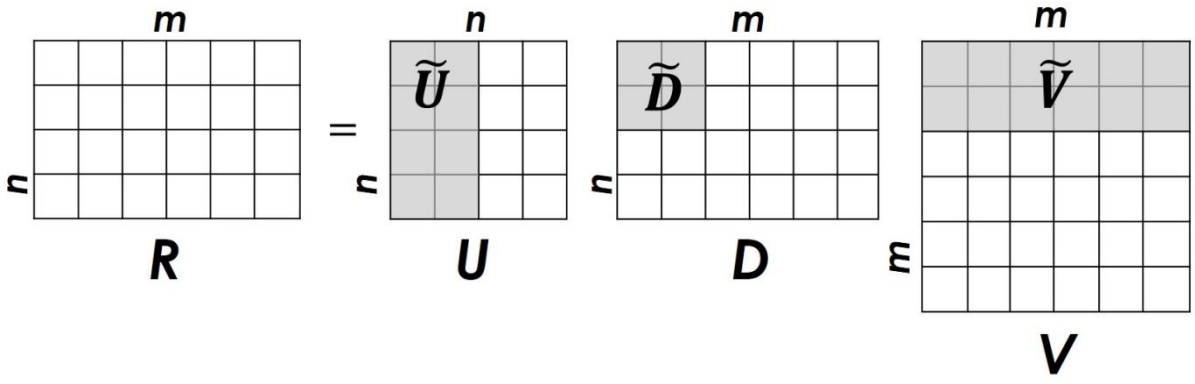


*Рисунок 1. SVD-разложение*

В полученном представлении матрица $\widehat{U}$ характеризует пользователей, а матрица $\tilde{V}$ - фильмы. Т.е. чтобы узнать оценку, данную *i*-м пользователем *j*-ому фильму, необходимо умножить *i*-ую строчку $\widehat{U}$ на *j*-ый столбец $\tilde{V}$. Таким образом, теперь каждый пользователь характеризуется не $m$-мерным вектором оценок, а *k*-мерным вектором условных параметров. Аналогично, каждый фильм теперь характеризуется не $n$-мерным вектором оценок всех пользователь, а *k*-мерным вектором условных параметров.

Перейдем к непосредственному описанию вычислительного алгоритма определения новой оценки фильма с идентификатором *id_f* для активного пользователя с идентификатором *id_p* на основе описанного метода SVD:

1) Задать параметр *k*.
2) Найти матрицы *U, D, V* сингулярного разложения матрицы *R*.
3) Отбросить малые значения сингулярных чисел матрицы $D$ и получить матрицы $\widetilde{D}, \widetilde{U}, \tilde{V}$ размерностей $\widetilde{D}$ - $k \times k$, $\widetilde{U}$ - $n \times k$, $\tilde{V}$ – $m \times m$.
4) Найти приближение исходной матрицы *R* матрицей $\tilde{R} = \widehat{U}\tilde{V}$,

где $\widehat{U} = \widetilde{U}\widetilde{D}$.

5) Определить оценку *id_p*-ого пользователя *id_f*-ого фильма как округленное произведение *id_p*-ой строки матрицы $\widehat{U}$ на *id_f*-ый столбец матрицы $\tilde{V}$:

$$\hat{r}_{\text{id\_p,id\_f}} = round\left(\hat{u}_{id\_p} \cdot \tilde{v}_{id\_f}\right)$$

6) Если полученная оценка вышла за пределы отрезка [1; 5], то необходимо округлить (увеличить или уменьшить) ее до ближайшей границы этого отрезка.

# Глава 3. Практические результаты

Данные, на которых проводилось практическое исследование, взяты с сайта [14]. В них содержится 100000 оценок 963 пользователей для 1682 фильмов. Для анализа работы программ, реализующих разные алгоритмы с разными коэффициентами, эти данные делятся на тренировочные и тестовые в разных пропорциях в процентах: 90/10, 95/5, 98/2. Данные представляют собой набор строк следующего типа: идентификатор пользователя, идентификатор фильма, оценка, данная этим пользователем этому фильму, а также временной штамп, когда эта оценка была поставлена. В рамках данной работы временные штампы не учитываются и не хранятся в данных.

Стоит также отметить, что при реализации метода *k*-ближайших соседей оценки прогнозируются последовательно для группы пользователей, а в методе SVD прогнозируются сразу все недостающие оценки в матрице оценок *R*.

Рассматриваемые подходы к построению рекомендаций реализованы на языке Python 2.0 в веб-приложении Jupyter Notebook, основным преимуществом которого является возможность хранить вместе код программ (можно с разбиением на части), заметки, графики и изображения. Разбиение кода на определенные части удобно при тестировании программы и многократном ее запуске с разными значениями параметров: достаточно запустить только часть ячеек, что существенно экономит время выполнения тестов.

Все результаты приведены при реализации указанных алгоритмов на ноутбуке Acer Aspire V с Intel Core i5-4210U, 1.7GHz с Turbo Boost до 2.7GHz.

Перед непосредственной реализацией методов построения новых оценок пользователей, проведен некоторый анализ данных, в ходе которого были найдены следующие характеристики:

- степень разреженности матрицы $R$ – $100000/(963 * 1682) = 0.06$;
- число оценок каждого пользователя – *num_nz* = [272, 62, 54, …, 0];
- среднее число оценок всех пользователей – 103.84;
- число пользователей, имеющих от 200 оценок (действующие пользователи) – 148;
- число пользователей, имеющих до 25 оценок (пассивные пользователи) – 141;
- средняя оценка по каждому пользователю – *av_rate* = [3.6, 3.7, 2.8, …, 0];
- средняя оценка по всем пользователям – *av_rates* = 3.5.

Для анализа работы алгоритмов было рассмотрено три характеристики: точность прогноза, полнота прогноза и время работы программы.

В следующей таблице приведены значения точности прогноза при использовании метода *k*-ближайших соседей с разными мерами сходства и при разбиении исходного множества данных на тренировочные и тестовые в соотношении 98/2:

*Таблица 1. Точность прогноза метода k-ближайших соседей, разбиение 98/2*

| **Число k** | **Эвклидова мера** | **Косинусная мера** | **Мера Пирсона** | **Модиф. Эвклида** |
|---|---|---|---|---|
| 1 | 78.03 | 80.49 | 80.37 | 79.26 |
| 4 | 82.18 | 83.87 | 83.7 | 83.34 |
| 7 | 83.39 | 84.54 | 84.57 | 84.46 |
| 10 | 83.44 | 85 | 84.84 | 84.56 |
| 13 | 83.99 | 85.03 | 84.89 | 84.79 |
| 16 | 84.06 | 85.07 | 85.04 | 84.89 |
| 19 | 84.16 | 85.05 | 85.02 | 84.91 |
| 22 | 84.28 | 85.06 | 85.09 | 85.16 |
| 25 | 84.48 | 85.03 | 85.17 | 85.21 |

При анализе точности прогноза и времени построения рекомендаций (представлено далее для рассматриваемых мер) было установлено, что мера Эвклида дает худшую точность, но при этом время, затрачиваемое на построение рекомендаций сильно меньше. В связи с этим рассматривается модифицированная мера Эвклида, которая находится с помощью вспомогательной функции, основанной на эвклидовом расстоянии:

$$\tilde{\rho}(p_1, p_2) = \sqrt{\sum_{i=1,\dots,m:\, r_{1,i}\cdot r_{2,i}\neq 0} \left(r_{1,i} - r_{2,i}\right)^2}.$$

Сама модифицированная мера Эвклида определяется следующим образом:

$$\mu(p_1, p_2) = \frac{1}{1+\tilde{\rho}(p_1,p_2)}.$$

Модифицированная мера Эвклида отличается от классической тем, что сравнение пользователей происходит только с учетом фильмов, которым оба пользователя поставили оценки.

Полнота прогнозов для разных значений *k* и разных мер одинаковая для такого разбиения данных: *recall*=0.997, всего 6 оценок не удалось определить указанным методом.

Время выполнения программы примерно одинаковое для разных значений *k*, но существенно отличается при использовании разных мер сходства: при применении меры Эвклида получается результат за 335 с, косинусной меры – за 690 с, меры Пирсона – за 870 с, модифицированной меры Эвклида – за 270 с.

Модифицированная мера Эвклида дает лучшее соотношение характеристик время/точность, при $k$ = 25 дает лучший результат из рассматриваемых вариантов, в связи с чем при дальнейшем анализе метода *k*-ближайших соседей будет использована именно модифицированная мера Эвклида.

В таблице 2 представлена точность метода *k*-ближайших соседей с использованием модифицированной меры Эвклида при разном разбиении исходных данных на тренировочные и тестовые:

*Таблица 2. Точность прогноза метода k-ближайших соседей*

| **Число k** | **Разбиение 90/10** | **Разбиение 95/5** | **Разбиение 98/2** |
|---|---|---|---|
| 1 | 78.98 | 79.21 | 79.26 |
| 10 | 84.15 | 84.36 | 84.56 |
| 20 | 84.64 | 84.8 | 85.05 |
| 30 | 84.77 | 84.88 | 85.27 |
| 40 | 84.78 | 84.96 | 85.3 |
| 50 | 84.8 | 84.92 | 85.26 |
| 60 | 84.77 | 84.96 | 85.23 |
| 70 | 84.75 | 84.95 | 85.14 |
| 80 | 84.65 | 84.84 | 85.09 |

Полнота прогноза: при разбиении 90/10 - *recall*=0.999, при разбиении 95/5 - *recall*=0.998, при разбиении 98/2 - *recall*=0.997. Количество оценок, которые не удалось спрогнозировать с использованием данного метода: при разбиении 90/10 - 12, при разбиении 95/5 - 10, при разбиении 98/2 – 6.

Время прогноза: при разбиении 90/10 - 1175 с, при разбиении 95/5 - 630 с, при разбиении 98/2 - 270 с.

Так как основное время в программе тратится на поиск группы наиболее похожих пользователей из всех пользователей, то для экономии времени будем производить поиск этой группы только из числа действующих пользователей (пользователей, которые поставили от 200 оценок) – т.е. выделим некоторую базу пользователей, из которых будем производить отбор наиболее похожих пользователей. В таблице 3 представлены значения точности прогнозов при разном разбиении:

*Таблица 3. Точность прогноза метода k-ближайших соседей с выделенной базой*

| **Число k** | **Разбиение 90/10** | **Разбиение 95/5** | **Разбиение 98/2** |
|---|---|---|---|
| 1 | 81.08 | 81.08 | 81.87 |
| 10 | 84.78 | 84.74 | 85.25 |
| 20 | 84.78 | 84.89 | 85.44 |
| 30 | 84.7 | 84.84 | 85.29 |
| 40 | 84.73 | 84.82 | 85.25 |
| 50 | 84.65 | 84.7 | 85.05 |
| 60 | 84.62 | 84.62 | 85 |
| 70 | 84.53 | 84.64 | 84.87 |
| 80 | 84.46 | 84.63 | 84.84 |

Из таблицы 3 видно, что максимальная точность во всех трех случаях разбиения множества достигается при $k = 20$.

Полнота прогноза: при разбиении 90/10 - *recall*=0.997, при разбиении 95/5 - *recall*=0.997, при разбиении 98/2 - *recall*=0.996. Количество оценок, которые не удалось спрогнозировать с использованием данного метода: при разбиении 90/10 - 26, при разбиении 95/5 - 14, при разбиении 98/2 – 9.

Время прогноза: при разбиении 90/10 - 390 с, при разбиении 95/5 - 220 с, при разбиении 98/2 - 102 с.

Если сравнить характеристики работы метода при поиске группы наиболее похожих пользователей из всех пользователей и только из группы действующих пользователей, получаются следующие результаты:

- при разбиении 90/10

*Таблица 4. Результаты*

| Метод | Макс. точность | Полнота | Время |
|---|---|---|---|
| k-ближайших соседей | 84.8 | 0.999 | 1175 |
| k-ближайших соседей с выделенной базой | 84.78 | 0.997 | 390 |

- при разбиении 95/5

*Таблица 5. Результаты*

| Метод | Макс. точность | Полнота | Время |
|---|---|---|---|
| k-ближайших соседей | 84.96 | 0.998 | 630 |
| k-ближайших соседей с выделенной базой | 84.89 | 0.997 | 220 |

- при разбиении 98/2

*Таблица 6. Результаты*

| Метод | Макс. точность | Полнота | Время |
|---|---|---|---|
| k-ближайших соседей | 85.3 | 0.997 | 270 |
| k-ближайших соседей с выделенной базой | 85.44 | 0.996 | 102 |

Из таблиц 4-6 видно, что точность метода *k*-ближайших соседей с выделенной базой примерно такая же, как точность метода *k*-ближайших соседей без выделенной базы, при этом время, затрачиваемое на построение прогнозных оценок с использованием второго метода, почти в 3 раза меньше. Таким образом, целесообразно пользоваться именно методом *k*-ближайших соседей с выделенной базой.

Перейдем к анализу классического метода SVD по тем же трем характеристикам (точность и полнота прогноза, время выполнения программы). В таблице 7 представлены значения точности рекомендаций при разном разбиении исходных данных и разном ранге приближенной матрицы (*k*).

*Таблица 7. Точность SVD*

| Число k | Разбиение 90/10 | Разбиение 95/5 | Разбиение 98/2 |
|---|---|---|---|
| 5 | 59.05 | 59.69 | 60.91 |
| 10 | 60.45 | 61.2 | 62.37 |
| 20 | 60.46 | 61.32 | 62.49 |
| 50 | 57.02 | 57.92 | 59.23 |
| 100 | 52.9 | 53.2 | 54.26 |

Как видно из таблицы 7, максимальная точность прогноза достигается при $k = 20$ во всех трех случаях разбиения исходных данных, что существенно уменьшает размерности матриц сингулярного разложения матрицы оценок $R$.

Математически исходная задача ставилась следующим образом:

$$\min_{err<20\%} (\alpha * \mathrm{err} + \beta * \mathrm{t}).$$

Рассматриваемый метод SVD не удовлетворил дополнительному условию ограничения на размер ошибки, поэтому в данном случае минимум достигается при использовании метода $k$-ближайших соседей, независимо от значений коэффициентов $\alpha, \beta$.

Полнота прогноза при использовании метода SVD – 1.0, так как прогнозируются все оценки.

При использовании метода SVD время прогноза существенно отличается для разного числа $k$, так как основное время в программе тратится на построение сингулярного разложения, которое в свою очередь строится с учетом этого параметра. В таблице 8 представлены значения затрачиваемого времени (в секундах) на построение прогноза:

*Таблица 8. Время SVD*

| Число k | Разбиение 90/10 | Разбиение 95/5 | Разбиение 98/2 |
|---|---|---|---|
| 5 | 4.91 | 5.12 | 5.09 |
| 10 | 9 | 9.48 | 9.72 |
| 20 | 13.23 | 13.9 | 14.47 |
| 50 | 17.73 | 18.59 | 19.33 |
| 100 | 22.52 | 23.55 | 24.28 |

Из таблицы 8 видно, что сингулярное разложение находится тем быстрее, чем больше нулевых элементов в матрице, т.е. чем меньше тренировочных данных.

Метод *k*-ближайших соседей дает гораздо лучшую точность, чем метод SVD, но время построения рекомендаций метода *k*-ближайших соседей гораздо больше даже при выделении базы действующих пользователей. Скорость работы метода *k*-ближайших соседей можно дополнительно увеличить путем параллельных вычислений, ведь каждая оценка строится независимо.

В данной задаче рекомендации фильмов, метод сингулярного разложения матрицы в чистом виде не применим, так как дает слишком маленькую точность, но его модификации (например, добавление корректирующих слагаемых) могут дать хорошие результаты как по точности, так и по полноте и времени.

# Заключение

Таким образом, в данной работе рассмотрено несколько подходов к решению центральной задачи разработки рекомендательной системы – прогнозированию оценок.

Метод *k*-ближайших соседей дал необходимую точность, для экономии времени построения прогноза и сохранении точности была рассмотрена модифицированная мера Эквлида, а также выделена база действующих пользователей, среди которых производился поиск группы наиболее похожих пользователей.

Метод SVD не дал необходимой точности, но при этом строил большое количество прогнозных оценок за очень короткое время. Модификации метода сингулярного разложения матрицы и его главная идея в приближении матрицы оценок некоторыми другими матрицами могут дать хорошие результаты при условии увеличения точности прогноза.

В будущем планируется полностью построить рекомендательную систему, добавить возможность задавать желаемые характеристики объектов, а также создать удобный интерфейс.

# Выводы

В ходе проведенных в работе исследований получены следующие основные результаты, которые выносятся на защиту:

1) Показано, что метод приближения матрицы оценок ее сингулярным представлением уменьшенной размерности (метод SVD) не применим к рассматриваемой задаче, поскольку дает большую ошибку (более 20%).
2) Проведен анализ применимости метода *k*-ближайших соседей: установлено, что он дает наилучший результат с коэффициентом $k$=20 среди рассмотренных вариантов.
3) Предложена модифицированная реализация метода *k*-ближайших соседей для экономии времени вычислений. Она выполнена с использованием модифицированной меры Эвклида с выделением базы действующих пользователей, среди которых производится отбор группы наиболее похожих вариантов.

# Список литературы


[1] Т. Сегаран, Программируем коллективный разум, СПб.: Символ-Плюс, 2008.

[2] F. Ricci, L. Rokach, B. Shapira and P. B. Kantor, Recommender Systems Handbook, LLC: Springer Science+Business Media, 2011.

[3] Д. Е. Королева и М. В. Филиппов, «Анализ алгоритмов обучения коллаборативных рекомендательных систем,» *Инженерный журнал: наука и инновации,* № 6, 2013.

[4] К. В. Воронцов, «Лекции по метрическим алгоритмам классификации,» 2008.

[5] А. И. Жданов, «Введение в вычислительную линейную алгебру,» Самара, 2011.

[6] А. Б. Нугуманова, И. А. Бессмертный, П. Пецина и Е. М. Байбурин, «Обогащение модели bag-of-words семантическими связями для повышения качества классификации текстов предметной области,» *Программные продукты и системы / Software & Systems*, № 2, 2016.

[7] В. В. Стрижов, *"Информационное моделирование". Конспект лекций.*.

[8] «Хабрахабр. Рекомендательные системы: You can (not) advise,» 23. 04. 2013. [В Интернете]. Available: https://habrahabr.ru/post/176549/. [Дата обращения: 21. 04. 2017].

[9] M. Howe, *Pandora's Music Recommender*.

[10] «Коллаборативная фильтрация,» [В Интернете]. Available: http://intellect.ml/kollaborativnaya-filtratsiya-4778/. [Дата обращения: 02.

04. 2017].

[11] «Хабрахабр. Коллаборативная фильтрация,» 28. 08. 2012. [В Интернете]. Available: https://habrahabr.ru/post/150399/. [Дата обращения: 25. 02. 2017].

[12] М. Кендалл и А. Стьюарт, Статистические выводы и связи, Москва: "Наука", 1973.

[13] А. Е. Умнов, Аналитическая геометрия и линейная алгебра, М.: МФТИ, 2011.

[14] «Grouplens,» [В Интернете]. Available: URL:https://grouplens.org/. [Дата обращения: 17. 02. 2016].

# Приложение

Ниже представлен код программы, реализующей описанные в данной работе методы.

Загрузка необходимых библиотек:

```
import graphlab as gl
import math
import os
import numpy as np
from graphlab import SFrame
import pandas as pd
from graphlab import SArray
import time
import scipy.sparse as sp
from scipy.sparse.linalg import svds
from math import sqrt
```

Функция *my_err* считает описанную в данной работе ошибку:

```
def my_err(pred, truth):
    pred = pred[truth.nonzero()].flatten()
    truth = truth[truth.nonzero()].flatten()
    e=np.zeros(pred.size);
    for k in range(pred.size):
        e[k]=abs(round(pred[k])-truth[k])/5.
return sum(e)/(pred.size)*100
```

Следующий фрагмент программы считывает данные, взятые из файла *Data.csv*, преобразует их, делит на тренировочные и тестовые, записывает в *numpy array* и определяет индексы оценок, которые необходимо будет спрогнозировать:

```
sf = gl.SFrame.read_csv('Data.csv',  column_type_hints={'num_p': np.integer})
```

```
num_p=963
num_f=1682
del sf["time_stamp"]
sf_train, sf_test = sf.random_split(.95, seed=5)
n_tr=gl.SFrame.to_numpy(sf_train)
n_test=gl.SFrame.to_numpy(sf_test)
#make matrix for train
rate_tr=np.zeros((num_p, num_f))
for k in range(n_tr.shape[0]):
   rate_tr[n_tr[k][0]-1][n_tr[k][1]-1]=n_tr[k][2]
#make matrix for test
rate_test=np.zeros((num_p, num_f))
for k in range(n_test.shape[0]):
   rate_test[n_test[k][0]-1][n_test[k][1]-1]=n_test[k][2]
ids= np.transpose(np.nonzero(rate_test))
```

Функция *measure* определяет меру сходства двух пользователей разными способами, в зависимости от аргумента *num_m*, отвечающего за номер используемой меры: 1 – мера Эвклида, 2 – косинусная мера, 3 – мера Пирсона, 4 – модифицированная мера Эвклида. Эта функция используется при реализации метода *k*-ближайших соседей:

```
def measure(num_m, x, y):
   res=0;
   if (num_m==1): #evklidean
      for i in range(len(x)):
            res+=(x[i]-y[i])**2
      res=1./(1+res**(0.5))
   if (num_m==2): #cosine
      for i in range(len(x)):
         res=res+x[i]*y[i]
      res=res/norma(x)/norma(y)
   if (num_m==3): #Pirson
      res, sumx, sumy=[0,0,0]
      sq_sumx=0; sq_sumy=0; p_sum=0
```

```
        num, den=[0,0]
        n=0.0
        n=len(x)
        for i in range(n):
            sumx=sumx+x[i]
            sumy=sumy+y[i]
            sq_sumx=sq_sumx+x[i]**2
            sq_sumy=sq_sumy+y[i]**2
            p_sum=p_sum+x[i]*y[i]
        num=p_sum-sumx*sumy/n
        den=((sq_sumx-sumx**2/n)*(sq_sumy-sumy**2/n))**(0.5)
        if den==0:
            res=0
        else:
            res=num/den
    if (num_m==4): #mod Evklidean
        for i in range(len(x)):
            if x[i]*y[i]!=0:
                res+=(x[i]-y[i])**2
        res=1./(1+res**(0.5))
    return res
```

Следующий фрагмент программы находит основные характеристики данных и выделяет базу действующих пользователей:

```
num_nz=np.zeros(num_p)
av_rate=np.zeros(num_p)
for k in range(num_p):
    num_nz[k]=int(rate_tr[k].nonzero()[0].shape[0])
    if num_nz[k]!=0:
        av_rate[k]=sum(rate_tr[k])/num_nz[k]
    else:
        av_rate[k]=0
av_num=sum(num_nz)/int(num_nz.shape[0])
num_act=int(num_nz[num_nz>200].shape[0])
num_pas=int(num_nz[num_nz<25].shape[0])
```

```
av_rates=sum(av_rate)/num_p
base=np.zeros((num_act, num_f))
p=0
for k in range(num_p):
   if num_nz[k]>200:
      base[p]=rate_tr[k]
      p+=1
```

Далее представлена основная часть программы метода *k*-ближайших соседей. Переменная *num_m* отвечает за номер использованной меры, *numK* – за количество наиболее похожих пользователей, *num_pred* – за количество прогнозных оценок:

```
num_m=4
for numK in np.arange(1, 85, 100):
   err=0
   num_nonpred=0
   num_pred=int(ids.shape[0])
   print 'num_pred='+str(num_pred)
   print 'numK='+str(numK)
   start_time=time.time()
   for num_it in range(num_pred):
      #predicting rate
      rate_pred=0
      id_p=ids[num_it][0]
      id_f=ids[num_it][1]
      mu_s=[[i,0.] for i in range(num_p)]
      for i in range(num_p):
         if rate_tr[i][id_f]!=0:
            #print 'find_id='+str(i)
            mu_s[i]=[i, measure(num_m, rate_tr[id_p],rate_tr[i])]
      mu_s=sorted(mu_s, key=lambda mu: mu[1])
      mu_s=mu_s[::-1]
      sum_mu=0
      for i in range(numK):
```

```
            rate_pred+=mu_s[i][1]*rate_tr[mu_s[i][0]][id_f]
            sum_mu+=mu_s[i][1]
        if sum_mu==0:
            rate_pred=0
            num_nonpred+=1
            #print "Couldn't predict rate for id_p="+str(id_p)+' and id_f='+str(id_f)
        else:
            rate_pred=round(rate_pred/sum_mu)
        #print 'rate_pred='+str(rate_pred)
        err+=abs(rate_pred-rate_test[id_p][id_f])
    err=err/5./num_pred*100
    pred_time=time.time()-start_time
    recall=(num_pred-num_nonpred)*1./num_pred
    print 'Error of k-nearest='+str(err)
    print 'Precision='+str(100-err)
    print 'Time='+str(pred_time)
    print 'Num od nonpredict rates='+str(num_nonpred)
    print 'Recall='+str(recall)
```

Далее представлена основная часть программы, реализующая метод SVD, описанный в данной работе:

```
for numK in [5, 10, 20, 50, 100]:
#get SVD components from train matrix. Choose k.
    U, D, V = svds(rate_tr, k = numK)
    D_diag=np.diag(D)
    rate_pred = np.dot(np.dot(U, D_diag), V)
    for k in range(num_p):
        for j in range(num_f):
            rate_pred[k][j]=round(rate_pred[k][j])
            if rate_pred[k][j]>5:
                rate_pred[k][j]=5
                num_out+=1
            if rate_pred[k][j]<1:
                rate_pred[k][j]=1
                num_out+=1
```

```
deb_time=time.time()-start_time
error=my_err(rate_pred, rate_test)
prec=100-error
print 'k='+str(numK)
print 'SVD my err: ' + str(error)
print 'SVD my prec: ' + str(prec)
print 'Time: ' + str(deb_time)
print 'Num of out elements='+str(num_out)
```